In the claims

1. (currently amended) A system comprising:

one or more processors;

a base program executable by the one or more processors and having one or more breakpoints; and,

a probe program associated with each breakpoint, the probe program executed and interpreted by an interpreter, the interpreter running on ~~that is executable by the~~ one or more processors ~~via an interpreter~~, the probe program independent of an architecture of the one or more processors, and the probe program generated from source code written in a high-level language,

the probe program associated with each breakpoint being executed when the breakpoint is reached during execution of the base program.

2. (currently amended) The system of claim 1, further comprising the interpreter ~~to interpret the probe program associated with each breakpoint~~.

3. (original) The system of claim 1, wherein the base program has a first address space and the probe program associated with each breakpoint has a second address space different from the first address space.

4. (original) The system of claim 3, further comprising one or more probe expressions that address objects of the base program in the first address space and that are used by objects of the probe program in the second address space to communicate with the objects of the base program in the first address space.

5. (original) The system of claim 3, further comprising a high-level language compiler to compile the probe program from source code written in the high-level language to object code.

6. (original) The system of claim 5, wherein the high-level language compiler is able to seamlessly intermix variables of the base program in the first address space and variables of the probe program in the second address space.

7. (original) The system of claim 3, wherein the probe program associated with each breakpoint is generated from an abstract syntax tree (AST) used to switch between the first and the second address spaces.

8. (previously presented) The system of claim 7, wherein the AST has a plurality of nodes.

9. (original) The system of claim 7, wherein the AST is able to be serialized into an interim format and deserialized from the interim format to reconstruct the AST.

10. (original) The system of claim 1, wherein probe program associated with each breakpoint is able to pass user messages by manipulating a state of one of the probe program and base program.

11. (original) The system of claim 1, wherein probe program associated with each breakpoint is able to pass user messages by manipulating a stack of one of the probe program and base program.

12. (original) The system of claim 1, wherein the base program is written in a high-level language different than the high-level language in which the probe program associated with each breakpoint is written.

13. (original) The system of claim 1, wherein the base program is written in a high-level language that is identical to the high-level language in which the probe program associated with each breakpoint is written.

14. (currently amended) A method for constructing and using a probe program associated with a breakpoint of a base program comprising:

constructing an abstract syntax tree (AST) having a plurality of nodes;

representing objects of the base program with at least some of the nodes of the AST;

representing objects of the probe program with other of the nodes of the AST;

switching between a first address space of the objects of the base program and a second address space of the objects of the probe program by traversing the AST; and,

serializing the AST into an interim format and storing the AST as serialized into the interim format,

such that the probe program is ~~executable via~~ executed and interpreted by an interpreter, the interpreter running on one or more processors, and the probe program is independent of an architecture of one or more processors executing the base program.

15. (previously presented) The method of claim 14, further comprising deserializing the AST from the interim format to reconstruct the AST.

16. (cancelled)

17. (original) The method of claim 14, manipulating a stack of the base program to pass user messages.

18. (currently amended) An article of manufacture comprising:

a machine-readable medium; and,

means in the medium for probing a base program at a breakpoint thereof in a processor architecture-independent manner, an abstract syntax tree (AST) constructed by ~~the~~ a probe program, serialized into an interim format, and stored,

wherein the means comprises the probe program, the probe program executed and interpreted by an interpreter running on one or more processors, and the probe program is independent of an architecture of the one or more processors.

19. (previously presented) The article of claim 18, wherein the means is written in a high-level language, and employs the AST.

20. (previously presented) The article of claim 18, wherein the medium is a recordable data storage medium.